|  |  |
| --- | --- |
| SELECT codigooficina, ciudad  FROM oficina | db.oficina.find({}, {  codigooficina: 1, ciudad: 1, \_id: 0  });  db.oficina.find( {  codigooficina: 1, ciudad: 1, \_id: 0  }); |
| Select \* from oficina | db.oficina.find({});  db.oficina.find();  **find({})**  **Qué es: find es un método en MongoDB que se utiliza para consultar documentos en una colección.**  **Para qué sirve: Permite buscar y recuperar documentos que coincidan con un criterio específico.**  **Parámetro {}:**  **El parámetro dentro de find es un objeto de consulta (query) que define los criterios de búsqueda.**  **En este caso, {} es un objeto vacío, lo que significa que no se aplica ningún filtro.** |
| SELECT ciudad, telefono  FROM oficina  WHERE pais = 'España' | db.oficina.find(  { pais: 'España' },  { ciudad: 1, telefono: 1, \_id: 0 }  );  **1. ciudad: 1**  **Qué significa: Indica que el campo ciudad debe incluirse en el resultado.**  **Para qué sirve: Si un campo tiene el valor 1 en la proyección, MongoDB incluirá ese campo en los documentos devueltos.**  **2. telefono: 1**  **Qué significa: Indica que el campo telefono también debe incluirse en el resultado.**  **3. \_id: 0**  **Qué significa: Indica que el campo \_id debe excluirse del resultado.**  **Para qué sirve: Por defecto, MongoDB siempre incluye el campo \_id en los resultados. Si no lo necesitas, puedes excluirlo estableciendo su valor en 0.**  db.oficinas.find({ nombre: /^B/ });  db.oficinas.find({ nombre: /E/i });  db.oficinas.find({ nombre: /E/i }).limit(1); |
| SELECT ciudad, telefono  FROM oficina  WHERE pais like '%E%' | db.oficina.find({ pais: /E/ }, { ciudad: 1, telefono: 1, \_id: 0 });  db.oficina.find({ pais: /^E/ }, { ciudad: 1, telefono: 1, \_id: 0 });  db.oficina.find({ pais: /E$/ }, { ciudad: 1, telefono: 1, \_id: 0 });   |  |  | | --- | --- | | **/E/** | **Contiene la letra "E" en cualquier posición.** |  |  |  | | --- | --- | | **/^E/** | **Comienza con la letra "E".** |  |  |  | | --- | --- | | **/E$/** | **Termina con la letra "E".** | |
| Los productos de cada pedido | db.pedido.find({}, {"producto.nombre": 1}).limit(10); |
| SELECT puesto, nombre, apellido1, apellido2, email  FROM empleado e  WHERE e.idempleadojefe IS NULL | db.empleado.find(  {idempleadojefe: null },  { puesto: 1, nombre: 1, apellido1: 1, apellido2: 1, email: 1, \_id: 0 }  ); |
| SELECT nombre, apellido1, apellido2, puesto  FROM empleado e  WHERE e.puesto <> 'Representante Ventas'  $ne es distinto | db.empleado.find(  { puesto: { $ne: 'Representante Ventas' } },  { nombre: 1, apellido1: 1, apellido2: 1, puesto: 1, \_id: 0 }  ); |
| SELECT DISTINCT p.estado  FROM pedido p | db.pedido.distinct("estado"); |
| SELECT DISTINCT p.idpago  FROM pago p  WHERE YEAR(p.fechapago) = 2008 | db.pago.distinct("idpago", { fechapago: { $gte: new Date("2008-01-01"), $lt: new Date("2009-01-01") } });  **{ fechapago: { $gte: new Date("2008-01-01"), $lt: new Date("2009-01-01") } }**  **Qué hace: Filtra los documentos donde el campo fechapago esté dentro de un rango específico.**  **Operadores:**  **$gte: Significa "mayor o igual que" (greater than or equal to).**  **$lt: Significa "menor que" (less than).** |
|  | db.pago.distinct("idpago", { fechapago: { $gte: new Date("2008-01-01")} }); |
| SELECT p.codigo\_pedido, p.codigo\_cliente, p.fecha\_esperada, p.fecha\_entrega  FROM pedido p  WHERE p.fecha\_entrega > p.fecha\_esperada  $expr: expresion | db.pedido.find(  { $expr: { $gt: ["$fechaentrega", "$fechaesperada"] } },  { codigopedido: 1, codigocliente: 1, fechaesperada: 1, fechaentrega: 1, \_id: 0 }  ); |
| SELECT \*  FROM pedido p  WHERE p.estado = 'Rechazado' AND YEAR(p.fechapedido) = 2009 | db.pedido.find(  {  estado: "Rechazado",  fechapedido: {  $gte: ISODate("2009-01-01T00:00:00Z"),  $lt: ISODate("2010-01-01T00:00:00Z")  }  }  ); |
|  | db.pedido.find({  estado: "Rechazado",  $expr: {  $eq: [{ $year: "$fechapedido" }, 2009] // Filtra por año (en este caso, 2009)  }  }); |
|  | db.pedido.distinct("estado", {  estado: "Rechazado",  $expr: {  $eq: [{ $year: "$fechapedido" }, 2009] // Filtra por año (en este caso, 2009)  }  }); |
| SELECT \*  FROM pedido p  WHERE MONTH(p.fecha\_entrega) = 01 | db.pedido.find({  $expr: {  $eq: [{ $month: "$fecha\_entrega" }, 1] // Filtra por el mes 1 (enero)  }  }); |
| SELECT \*  FROM pago p  WHERE YEAR(p.fecha\_pago) = 2008 AND p.forma\_pago = 'PayPal' | db.pago.find({  $expr: {  $eq: [{ $year: "$fechapago" }, 2008] // Filtra por el año 2008  },  formapago: "PayPal" // Filtra por la forma de pago Paypal  }); |
| SELECT \*  FROM producto p  WHERE p.gama = 'Ornamentales' AND p.cantidadstock > 100  ORDER BY p.precioventa DESC | db.producto.find({  "gamaproducto.gama": "Ornamentales",  cantidadstock: { $gt: 100 } // Filtra por cantidadstock mayor que 100  }).sort({ precioventa: -1 }); // Ordena por precioventa de forma descendente  #### 1 ASC -1 DESC |
| SELECT \*  FROM cliente c  WHERE c.ciudad = 'Madrid' AND c.codigo\_empleado\_rep\_ventas IN (11, 30) | db.cliente.find({  ciudad: "Madrid",  idcodigoempleadoventas: { $in: [11, 30] }  }); |
| ##seleccion que campo se mostraran | db.cliente.find(  {  ciudad: "Madrid",  idcodigoempleadoventas: { $in: [11, 30] }  },  { nombrecontacto: 1, \_id: 0 }  ); |
| RESUMEN |  |
| SELECT COUNT(e.codigo\_empleado) as 'Cantidad de empleados'  FROM empleado e | db.empleado.aggregate([  {  $group: {  \_id: null,  "Cantidad de empleados": { $sum: 1 }  }  }  ]); |
|  | db.empleado.countDocuments(); |
| SELECT c.pais, COUNT(c.pais)  FROM cliente c  GROUP BY c.pais | db.cliente.aggregate([  {  $group: {  \_id: "$pais",  "Cantidad": { $sum: 1 }  }  },  {  $project: {  "\_id": 0,  "pais": "$\_id",  "Cantidad": 1  }  }  ]);   * **$group**: Agrupa los documentos por el campo "pais" y utiliza el operador **$sum** para contar cuántos documentos hay en cada grupo. * **$project**: Proyecta los campos deseados en el resultado final, renombrando "\_id" como "pais".   db.cliente.distinct("pais").forEach(function(pais) {  var count = db.cliente.countDocuments({ pais: pais });  print(pais + ": " + count);  }); |
| SELECT AVG(p.total) AS 'Promedio de pago en 2009'  FROM pago p  WHERE YEAR(p.fecha\_pago) = 2009 | * **$match**: Filtra los documentos para seleccionar solo aquellos que tengan fechas de pago dentro del año 2009. * **$group**: Agrupa los documentos en un solo grupo (usando **\_id: null**) y calcula el promedio de los valores del campo "total" utilizando el operador **$avg**. |
| db.collection.aggregate([  { $match: { campo: valor } },  { $group: { \_id: "$campo", total: { $sum: "$cantidad" } } }  ]);  **find()**: Es un método utilizado para recuperar documentos de una colección que coinciden con un criterio de consulta especificado. **find()** es más adecuado para consultas simples que no requieren transformación o manipulación de datos complejos. Puedes usar **find()** para buscar documentos que coincidan con ciertos campos o condiciones.  **aggregate()**: Es un método de la colección que realiza operaciones de agregación de datos en los documentos de la colección. Permite realizar transformaciones avanzadas y cálculos en los datos recuperados, como agrupaciones, sumas, promedios, filtrados, y otras operaciones. **aggregate()** es útil cuando necesitas realizar operaciones de datos más complejas que van más allá de una simple búsqueda. | db.pago.aggregate([  {  $match: {  fechapago: {  $gte: new Date("2009-01-01"),  $lt: new Date("2010-01-01")  }  }  },  {  $group: {  \_id: null,  "Promedio de pago en 2009": { $avg: "$total" }  }  }  ]);  **2. Etapa 1: $match**  **3. Etapa 2: $group**  ** \_id: null: Agrupa todos los documentos en un único grupo (sin partición por categoría).**  ** "Promedio de pago en 2009": Especifica un nuevo campo en la salida que contendrá el promedio de los valores del campo total.**  ** $avg: Operador de agregación para calcular el promedio.** |
| SELECT MIN(p.precio\_venta) AS 'Producto mas barato', MAX(p.precio\_venta) AS 'Producto mas caro'  FROM producto p | db.producto.aggregate([  {  $group: {  \_id: null,  "Producto mas barato": { $min: "$precioventa" },  "Producto mas caro": { $max: "$precioventa" }  }  }  ]); |
| SELECT COUNT(c.codigo\_cliente) AS 'Clientes de Madrid'  FROM cliente c  WHERE c.ciudad = 'Madrid' | db.cliente.aggregate([  {  $match: {  ciudad: 'Madrid'  }  },  {  $group: {  \_id: null,  'Clientes de Madrid': { $sum: 1 }  }  }  ]); |
| SELECT COUNT(c.codigo\_cliente)  FROM cliente c  WHERE ciudad LIKE 'M%' | db.cliente.aggregate([  {  $match: {  ciudad: /^M/  }  },  {  $group: {  \_id: null,  'Clientes que empiezan con M': { $sum: 1 }  }  }  ]); |
| SELECT c.codigo\_empleado\_rep\_ventas AS 'Id Empleado', COUNT(c.codigo\_empleado\_rep\_ventas) AS 'Cantidad de Clientes'  FROM cliente c  GROUP BY c.codigo\_empleado\_rep\_ventas | db.cliente.aggregate([  {  $group: {  \_id: "$codigo\_empleado\_rep\_ventas",  "Cantidad de Clientes": { $sum: 1 }  }  },  {  $project: {  "\_id": 0,  "Id Empleado": "$\_id",  "Cantidad de Clientes": 1  }  }  ]); |
| * La etapa **$group** agrupa los documentos por el campo "codigo\_empleado\_rep\_ventas" y utiliza el operador **$sum** para contar la cantidad de documentos en cada grupo. * La etapa **$project** se utiliza para cambiar el nombre del campo **\_id** a "Id Empleado" y proyectar solo los campos necesarios en el resultado final. |  |
| SELECT o.ciudad, COUNT(e.codigooficina) AS 'Cantidad de empleados'  FROM oficina o  INNER JOIN empleado e on e.idoficina=o.idoficina  GROUP BY o.ciudad  SELECT c.nombrecliente, COUNT(p.codigocliente) AS 'Cantidad de pedidos realizados' FROM cliente c LEFT JOIN pedido p ON c.idcliente = p.idcliente GROUP BY c.nombrecliente   * db.cliente.aggregate([  {  $lookup: {  from: "pedido",  localField: "codigocliente",  foreignField: "idcliente",  as: "pedidos"  }  },  {  $unwind: {  path: "$pedidos",  preserveNullAndEmptyArrays: true  }  },  {  $group: {  \_id: "$nombrecliente",  cantidadPedidos: { $sum: 1 }  }  },  {  $project: {  \_id: 1,  nombrecliente: "$id",  cantidadPedidos: 1  }  } ]); | Etapa 1: $lookup   **Objetivo:** Hacer un *join* entre la colección oficina y la colección empleado.   **Claves importantes:**   * **from: "empleado"**: Indica la colección secundaria (empleado) que queremos unir. * **localField: "idoficina"**: Campo en la colección oficina que se utiliza como clave de relación. * **foreignField: "idoficina"**: Campo en la colección empleado que corresponde a la clave de relación. * **as: "empleados"**: Nombre del campo en el que se guardará el resultado del *join*. Será un arreglo que contiene todos los documentos de empleado relacionados con cada oficina.   Etapa 2: $unwind   **Objetivo:** Descomponer el arreglo empleados en documentos individuales.   **Efecto:** Por cada elemento en el arreglo, se crea un nuevo documento.  Etapa 3: $group   **Objetivo:** Agrupar los documentos por el campo ciudad y calcular la cantidad de empleados en cada ciudad.   **Claves importantes:**   * **\_id: "$ciudad"**: Define la clave de agrupación, en este caso, la ciudad. * **$sum: 1**: Suma 1 por cada documento en el grupo.   Etapa 4: $project   **Objetivo:** Formatear el resultado final para mostrar únicamente los campos necesarios.   **Claves importantes:**   * **\_id: 0**: Excluye el campo \_id generado automáticamente. * **ciudad: "$\_id"**: Renombra el campo \_id a ciudad. * **cantidadEmpleados: 1**: Incluye el campo cantidadEmpleados en la salida.   **Resumen del flujo:**   1. Realiza un *join* para unir las colecciones oficina y empleado basándose en el campo idoficina. 2. Convierte los arreglos de empleados en documentos individuales con $unwind. 3. Agrupa los documentos por ciudad y calcula la cantidad de empleados en cada ciudad. 4. Ajusta el formato del resultado final con $project. |